Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Nome completo do Autor

Título da monografia
se for longo ocupa esta linha também

São Paulo
Dezembro de 2015

Título da monografia
se for longo ocupa esta linha também

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Nome do Supervisor
[Cosupervisor: Prof. Dr. Nome do Cosupervisor]

São Paulo
Dezembro de 2015

Resumo

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave).

Palavras-chave: palavra-chave1, palavra-chave2, palavra-chave3.

Abstract

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa.

Keywords: keyword1, keyword2, keyword3.

Sumário

Capítulo 1

Introdução

Uma monografia deve ter um capítulo inicial que é a Introdução e um capítulo final que é a Conclusão. Entre esses dois capítulos poderá ter uma sequência de capítulos que descrevem o trabalho em detalhes. Após o capítulo de conclusão, poderá ter apêndices e ao final deverá ter as referências bibliográficas.

Para a escrita de textos em Ciência da Computação, o livro de Justin Zobel, *Writing for Computer Science* (Zobel, 2004) é uma leitura obrigatória. O livro *Metodologia de Pesquisa para Ciência da Computação* de Wazlawick (2009) também merece uma boa lida.

O uso desnecessário de termos em lingua estrangeira deve ser evitado. No entanto, quando isso for necessário, os termos devem aparecer *em itálico*.

```
Modos de citação:
indesejável: [AF83] introduziu o algoritmo ótimo.
indesejável: (Andrew e Foster, 1983) introduziram o algoritmo ótimo.
certo : Andrew e Foster introduziram o algoritmo ótimo [AF83].
certo : Andrew e Foster introduziram o algoritmo ótimo (Andrew e Foster, 1983).
certo : Andrew e Foster (1983) introduziram o algoritmo ótimo.
```

Uma prática recomendável na escrita de textos é descrever as legendas das figuras e tabelas em forma auto-contida: as legendas devem ser razoavelmente completas, de modo que o leitor possa entender a figura sem ler o texto onde a figura ou tabela é citada.

Apresentar os resultados de forma simples, clara e completa é uma tarefa que requer inspiração. Nesse sentido, o livro de Tufte (2001), *The Visual Display of Quantitative Information*, serve de ajuda na criação de figuras que permitam entender e interpretar dados/resultados de forma eficiente.

Capítulo 2

Aprendizado de Reforço

Aprendizado de Reforço (AR) é uma das grandes áreas que compõe Aprendizado de Máquina (?), onde são estudados algoritmos que descrevem o comportamento de *agentes* em um *ambiente*, buscando o maior *retorno* por suas ações (ilustrado na figura 2.1).

Essa definição geral admite aplicação em uma grande quantidade de problemas (?), sendo limitada principalmente pela existência de um ambiente que possa ser eficientemente simulado durante o processo de aprendizado do algoritmo. Consequentemente, o aprendizado muitas vezes depende de uma quantidade grande de recursos computacionais.

Geralmente, os problemas da área são formulados como Processos de Decisão de Markov (MDP) finitos(?). Um Processo de Decisão de Markov é uma quíntupla (S, A, T, R, γ)(?), onde:

- S é o conjunto de estados (que representa o ambiente)
- A é o conjunto de ações que o agente pode tomar
- $T : S \times S \times A \to [0, 1]$ representa a probabilidade de um novo estado s' dados o estado atual s e ação a
- $R : S \times A \to \mathbb{R}$ a função de retorno

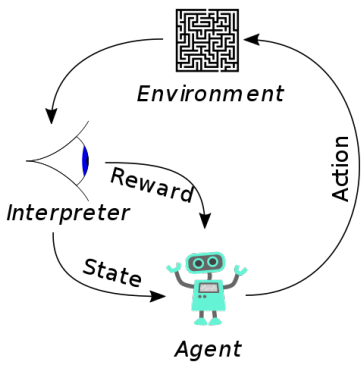

Figura 2.1: *Ilustração de Aprendizado de Reforço*

- $\gamma \in [0, 1]$ é um fator que determina o a importância de retornos futuros comparados com retornos imediatos.

Nesse trabalho, trataremos de um problema de *informação imperfeita*, onde o agente não poderá observar diretamente o estado s, só uma parte dele $o(s)$. Além disso, temos um conjunto de estados chamado de *estados terminais*, onde a probabilidade de transição para qualquer outro estado é 0.

O agente define uma *política* $\pi : A \times o(S) \to [0, 1]$ que representa a distribuição de probabilidade da escolha de uma ação em um dado estado $\pi(a|o(s))$.

Assim temos um mecanismo para gerar amostras, partindo de um estado S_0, selecionamos uma ação $A_0 \sim \pi(a|o(S_0))$, deixamos o agente observar $R_1 = R(S_0, A_0)$ e selecionamos um estado $S_1 \sim T(s|S_0, A_0)$, a partir do qual o processo se repete, até alcançar um estado S_T que seja terminal. Ao conjunto $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, ..., R_T, S_T$, damos o nome de *episódio*.

O objetivo dos nossos agentes é, através de suas observações dos episódios, aprender uma política que maximiza o *retorno descontado* G_1 onde: $G_t := R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ... = R_t + \gamma G_{t+1}$.

Será útil para nossa discussão definir a função *valor* $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ de um estado s, para a política π em um tempo t qualquer, e a função *ação-valor* $q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$.

Isso é porque muitos algoritmos de AR consistem em definir uma política, estimar a função valor ou ação-valor dessa política, e usá-lá para criar uma nova política, aumentando a probabilidade de visitar um estado quanto maior for o seu valor (nesse caso, é necessário também modelar o ambiente), ou aumentando a probabilidade de tomar ações quanto maior for a ação-valor.

Capítulo 3

Ambiente

O ambiente utilizado no trabalho foi o ELF (?). Criado para a pesquisa em IA, sua escolha propõe múltiplos benefícios: proporciona todos os desafios característicos de jogos RTS, dispensa das complexidades irrelevantes para pesquisa que estão presentes em jogos destinados para o mercado, é integrado com uma framework de Aprendizado de Reforço, baseada em PyTorch (?), faz um uso altamente eficiente dos recursos computacionais, e, por último, proporciona alto grau de controle para o usuário.

O jogo consiste de dois jogadores disputando por recursos, construindo unidades e as controlando com o objetivo final de destruir a base do adversário. Mais precisamente, cada jogador começa com sua base em cantos opostos de um mapa quadricular, que contém fontes de recursos. É importante ressaltar que os jogadores só tem visão do que está suficiente próximo de suas unidades (e conhecimento da localização da base inimiga), isso significa que se trata de um problema de *informação parcial*.

A observação do estado é uma discretização do mapa em uma grid 20x20, e a informação é distribuída em múltiplos canais (respeitando a área de visão), primeiro revelando a posição de unidades de cada tipo, depois o HP e por fim um canal que representa as fontes de recursos.

As ações disponíveis para os agentes são as mesmas em todos os estados, e estão descritas na tabela 3.1.

Comando	Descrição
INATIVO	Não faz nada
CONSTRÓI-TRABALHADOR	Se a base está inativa, constrói um trabalhador.
CONSTRÓI-QUARTEL	Move um trabalhador (coletando ou inativo) para um lugar vazio e constrói um quartel.
CONSTRÓI-GLADIADOR	Se existe um quartel inativo, constrói um gladiador.
CONSTRÓI-TANQUE	Se existe um quartel inativo, constrói um tanque.
BATER-E-CORRER	Se existem tanques, move eles em direção a base inimiga e ataca, fuja se forem contra atacados.
ATACAR	Gladiadores e tanques atacam a base inimiga.
ATACAR EM ALCANCE	Gladiadores e tanques atacam inimigos a vista.
TODOS A DEFESA	Gladiadores e tanques atacar tropas inimigas perto da base ou da fonte de recursos.

Tabela 3.1: *Ações*

Capítulo 4

Desenvolvimentos

Embora neste exemplo tenhamos apenas um capítulo, entre a introdução e a conclusão de uma monografia podemos ter uma sequência de capítulos descrevendo o trabalho e os resultados. Estes podem descrever fundamentos, trabalhos relacionados, método/modelo/algoritmo proposto, experimentos realizados, resulatdos obtidos.

Cada capítulo pode ser organizado em seções, que por sua vez pode conter subseções.

Um exemplo de figura está na figura 4.1.

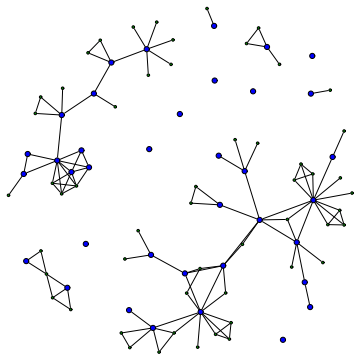

Figura 4.1: *Exemplo de uma figura.*

Capítulo 5

Conclusões

Texto texto[1].

[1]Exemplo de referência para página Web: www.vision.ime.usp.br/~jmena/stuff/tese-exemplo

Capítulo 6

Definições

Um grafo $G = (V, A)$ é um par ordenado, onde V e A são conjuntos disjuntos e cada elemento de A é um par não-ordenado de elementos de V. Os elementos de V são chamados de **vértices** e os de A são chamados de **arestas**.

Um grafo é dito **incremental** se suporta a adição de arestas. Analogamente, um grafo dinâmico é dito **decremental** se suporta a remoção de arestas. Um grafo incremental **ou** decremental é dito **parcialmente dinâmico**. Um grafo incremental **e** decremental é dito **totalmente dinâmico**.

Apêndice A

Título do apêndice

Texto texto.

Referências Bibliográficas

Tufte(2001) Edward Tufte. *The Visual Display of Quantitative Information.* Graphics Pr, 2nd edição. Citado na pág. 1

Wazlawick(2009) Raul S. Wazlawick. *Metodologia de Pesquisa em Ciencia da Computação.* Campus, primeira edição. Citado na pág. 1

Zobel(2004) Justin Zobel. *Writing for Computer Science: The art of effective communication.* Springer, segunda edição. Citado na pág. 1